BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Evaluating Retrieval-Augmented Generation Variants for Biomedical Multiple-Choice Question Answering

Sadia Sultana
200041134
Mosammat Zannatul Samarukh
200041139
Saiyma Sittul Muna
200041141

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

Evaluating Retrieval-Augmented Generation Variants for Biomedical Multiple-Choice Question Answering

Sadia Sultana
200041134
Mosammat Zannatul Samarukh
200041139
Saiyma Sittul Muna
200041141

Department of Computer Science and Engineering
Islamic University of Technology
September, 2025

Declaration of Candidates

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Sadia Sultana**, **Mosammat Zannatul Samarukh**, and **Saiyma Sittul Muna** under the supervision of **Tareque Mohmud Chowdhury**, Assistant Professor, Department of Computer Science and Engineering and co-supervision of **Ajwad Abrar Mostofa**, Junior Lecturer, Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Tareque Mohmud Chowdhury
Assistant Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: September 29, 2025

Ajwad Abrar Mostofa
Junior Lecturer
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: September 29, 2025

Sadia Sultana
Student ID: 200041134
Date: September 29, 2025

Mosammat Zannatul Samarukh
Student ID: 200041139
Date: September 29, 2025

Saiyma Sittul Muna
Student ID: 200041141
Date: September 29, 2025

Dedicated to our families, friends and teachers, whose encouragement and sacrifices have been our constant source of strength.

Contents

List of Figures

List of Tables

Abstract

The scarcity of reliable biomedical Question-Answering (QA) resources in low-resource languages like Bangla, spoken by over 230 million, limits access to accurate medical knowledge. Large Language Models (LLMs) like GPT and LLaMA often produce factually incorrect answers due to static training data, a critical issue in medical QA where accuracy is vital. To address this, we evaluated Retrieval-Augmented Generation (RAG) to combine external knowledge retrieval with generative reasoning, enhancing answer verifiability. We introduce BanglaMedQA, a curated dataset of 1,000 medical Multiple Choice Questions (MCQs) with rationales from Bangladeshi entrance exams (MBBS, BDS, AFMC) spanning from 1990–2024, and Bangla MMedBench, a translated MMedBench dataset for complex reasoning. Using a Bangla biology textbook corpus and web search, we tested RAG variants - Traditional, Zero-Shot Fallback, Agentic, Iterative Feedback, and Aggregate retrieval with LLMs. Agentic RAG achieved the highest accuracy (89.54% with openai/gpt-oss-120b), followed by Iterative RAG (88.73%) across the BanglaMedQA dataset, with every method outperforming Zero-Shot. Despite challenges in translation fidelity and retrieval robustness, this work enhances Bangla medical QA accuracy and accessibility.

Chapter 1

Introduction

1.1 Introductory Information

Medical Question Answering (QA) has emerged as a critical application of Natural Language Processing (NLP), enabling healthcare practitioners, students, and patients to access reliable information efficiently. Recent advances in Large Language Models (LLMs) such as GPT and LLaMA have demonstrated strong capabilities in generating coherent and contextually relevant responses. However, LLMs face two key limitations: their knowledge is bounded by training data, and their responses can be factually incorrect or "hallucinated" [1], [2].

Retrieval-Augmented Generation (RAG) addresses these challenges by combining external knowledge retrieval with generative reasoning, making it particularly valuable in domains like medicine where accuracy and verifiability are essential [3], [4].

While RAG has shown promising results in English and other high-resource languages, its effectiveness in low-resource languages remains underexplored. Bangla, spoken by over 230 million people, lacks high-quality biomedical datasets and benchmarks [5], [6], [7]. This gap hinders the development of reliable medical QA systems for Bangla-speaking communities, creating barriers in education and healthcare support.

1.2 Motivations and Scope

The motivation for this research arises from two interrelated needs:
(i) Enabling equitable access to biomedical knowledge for Bangla-speaking learners and practitioners.

(ii) Systematically evaluating advanced QA strategies in a low-resource language setting.

Despite the proliferation of multilingual models, their performance in Bangla medical reasoning remains unverified due to the absence of curated datasets and benchmarks [1]. This thesis seeks to bridge that gap by constructing two Bangla biomedical QA resources, BanglaMedQA and Bangla MMedBench and by evaluating multiple RAG strategies on them. By doing so, it lays the groundwork for reliable Bangla medical reasoning systems and contributes to the broader goal of democratizing healthcare AI for underrepresented languages.

1.3 Problem Statement

Although Retrieval-Augmented Generation (RAG) improves factuality in Question Answering (QA), its impact on Bangla biomedical QA remains untested. The lack of curated Bangla medical datasets and systematic evaluation frameworks prevents fair comparison of RAG variants and leaves open questions about their adaptability to low-resource settings. This research addresses the following problem:

> *How effective are different RAG variants in improving accuracy and reasoning quality for Bangla biomedical question answering compared to traditional and Zero-Shot approaches?*

1.4 Research Challenges

This research confronts several challenges:

- **Data Scarcity:** No existing large-scale Bangla biomedical Question Answering (QA) dataset with rationales [5], [6].
- **Translation Quality:** Preserving accuracy and medical meaning during dataset translation [8].
- **Retrieval Complexity:** Ensuring that relevant context is retrieved from textbooks or external sources in Bangla [3], [9].
- **Model Adaptability:** Assessing whether Retrieval-Augmented Generation (RAG) strategies designed for English generalize effectively to Bangla biomedical contexts [4].
- **Evaluation Rigor:** Utilizing robust metrics to measure not only answer correctness but also rationale quality [10].

1.5 Contributions

This thesis makes three key contributions:

1. **BanglaMedQA**: We curated the first Bangla medical Question Answering (QA) dataset derived from authentic admission test questions.
2. **Bangla MMedBench**: We developed a translated and preprocessed Bangla version of the MMedBench dataset, enabling reasoning-intensive evaluation in Bangla.
3. **Systematic Evaluation of Retrieval-Augmented Generation (RAG) Variants**: We implemented and benchmarked traditional RAG [3], RAG with Fallback, Agentic RAG [11], Iterative Feedback RAG [12], and Aggregate k-values RAG [4] on our BanglaMedQA dataset.

Together, these contributions provide a benchmark for Bangla biomedical QA and advance the understanding of RAG in low-resource medical Natural Language Processing (NLP). In the following sections, we present the related works, describe our proposed methodology, discuss the experimental results, and conclude with key findings and future directions.

Chapter 2

Related Works

2.1 Medical Question Answering (Medical QA)

Automated solutions for aggregating and summarizing information are becoming increasingly important for patients and healthcare professionals, given the rapid growth of online medical literature. Large Language Models (LLMs), with their sophisticated generative capabilities, have demonstrated strong potential in a variety of Natural Language Processing (NLP) applications [1], [2]. Biomedical Question Answering (QA) systems have the potential to revolutionize clinical workflows by retrieving precise and contextually relevant information, thereby improving efficiency, reducing cognitive load, and supporting evidence-based decision-making [13], [14].

However, medical QA remains highly challenging due to the complexity of biomedical terminology, the necessity for deep semantic understanding, and the lack of domain-specific adaptation in most general-purpose language. Biomedical language is filled with abbreviations, nuanced meanings, and specialized vocabulary that are often not well-covered by general corpora.

Broadly, biomedical QA tasks fall into two categories:

1. **Multiple-Choice Question Answering (MCQ):**

   In this format, models are provided with a question and a fixed set of answer options. Pal et al. introduced **MedMCQA**, a dataset of 193k multiple-choice questions from Indian medical exams, which has been widely used for model benchmarking [15]. Similarly, Jin et al. proposed **PubMedQA**, which also includes a multiple-choice variant for reasoning over PubMed articles [8].

2. **Generative Answering:**

In this setup, models generate free-form answers to biomedical questions. Tsatsaronis et al. presented the **BioASQ** challenge, which evaluates systems on biomedical semantic indexing and Question Answering (QA) based on PubMed literature [10]. Generative QA benchmarks like BioASQ enable testing deeper reasoning but are also more prone to hallucinations.

2.2 Low-Resource Languages in Medical NLP

For English, many datasets exist, but for Bangla such resources are scarce. Shafayat et al. introduced **BEnQA**, a bilingual benchmark of Science Question Answering (QA) in Bangla and English [5]. Other resources include the Bangla Medical Dataset on Kaggle [6] and a Bangla health Named-Entity Recognition (NER) dataset [7], [16], [17]. However, these do not address biomedical QA reasoning directly, leaving a major gap that our datasets aim to fill.

2.3 Retrieval-Augmented Generation (RAG)

Lewis et al. introduced Retrieval-Augmented Generation (RAG), which combines dense passage retrieval with generative models to ground responses in external knowledge sources [3], [18]. This approach mitigates limitations of static Large Language Models (LLMs) by providing factual grounding, reducing hallucinations, and enabling models to incorporate up-to-date knowledge. Izacard and Grave further showed that retrieval-based generation consistently outperforms parametric-only methods in knowledge-intensive Natural Language Processing (NLP) tasks [9]. In the medical domain, Zhang et al. benchmarked RAG systems on biomedical Question Answering (QA) and demonstrated substantial improvements in factual correctness and reasoning quality [4].

Despite these benefits, vanilla RAG pipelines face challenges such as retrieval errors, irrelevant passages, and instability in reasoning. To address these issues, several RAG variants have been proposed, each improving adaptability and robustness in different ways:

RAG with Fallback

Fallback-based RAG strategies combine retrieval with a Zero-Shot or parametric fallback mechanism. For example, Lewis et al. [3] and subsequent extensions demonstrated that when retrieval fails to yield high-quality passages, the system can revert

to generating an answer directly from the model's internal knowledge. This hybrid setup ensures coverage while reducing hallucinations, particularly useful in specialized domains like medicine where retrieval databases may be incomplete.

Agentic RAG

Agentic RAG frameworks view the Question Answering (QA) system as an "agent" capable of deciding when and how to retrieve knowledge [19]. Yao et al. proposed the **ReAct** framework, which allows models to interleave reasoning steps with retrieval and external actions [11]. This approach demonstrated improvements on reasoning-intensive QA tasks by dynamically routing between retrieval, external web search, or internal generation. Similarly, Shinn et al. introduced **Reflexion**, which equips models with feedback-driven self-improvement by reflecting on past responses and reformulating queries when retrieval was insufficient [12]. These methods highlight the growing shift towards agentic, self-directed retrieval strategies.

Iterative Feedback RAG

Iterative RAG approaches aim to refine retrieval through multiple feedback loops [20]. Instead of relying on a single retrieval step, the system reformulates queries, extracts keywords, and retrieves additional context in multiple iterations. Shinn et al.'s Reflexion [12] exemplifies this paradigm, showing that iterative reformulation reduces retrieval noise and enhances factual grounding. Such iterative designs are especially effective for biomedical Question Answering (QA), where precise terminology and nuanced meanings are critical.

Aggregate k-Values RAG

Another line of work explores aggregating information across multiple retrieval depths or values of k. In multiple studies, it has been demonstrated that combining evidence from several retrieved contexts improves stability and robustness of Retrieval-Augmented Generation (RAG) outputs, especially in distributed domains such as biomedical literature [4] [21]. By aggregating across multiple passages rather than relying on a single best retrieval, models are less sensitive to retrieval noise and achieve higher consistency.

Together, these studies demonstrate that augmenting vanilla RAG with fallback strategies, agentic decision-making, iterative refinement, and aggregation mechanisms can substantially improve Question Answering (QA) performance. These

innovations are particularly relevant in the biomedical domain, where factual correctness, interpretability, and reliability are critical.

2.4 Research Gap and Motivation

Despite notable progress in Natural Language Processing (NLP) and Large Language Models (LLMs), several gaps remain in the domain of biomedical Question Answering (QA), particularly in multilingual and resource-constrained contexts.

1. **Limited Biomedical QA in Bangla and Low-Resource Languages**
   Most existing biomedical QA systems and datasets are developed for English. Low-resource languages like Bangla remain underrepresented, limiting access for a large portion of practitioners, students, and patients in non-English speaking regions.

2. **Scarcity of Domain-Specific Datasets**
   Widely used datasets (e.g., MedQA [22], PubMedQA, MedMCQA) are predominantly in English and often based on Western curricula. There is a lack of curated, high-quality biomedical MCQ datasets tailored to South Asian or Bangla medical textbooks, creating a significant barrier for localized applications.

3. **Challenges in Applying General LLMs to Biomedical MCQ Tasks**
   Open-domain LLMs (e.g., GPT, LLaMA, Mistral) show strong general reasoning but often lack specialized biomedical knowledge [23]. Without domain adaptation, these models struggle with terminologies, abbreviations, and reasoning required for medical MCQs.

4. **Limitations of Current Retrieval-Augmented Generation (RAG) Approaches**
   While RAG has improved factual accuracy by grounding LLMs in external knowledge, most biomedical RAG studies focus on research articles and clinical notes, not structured MCQs. Existing RAG pipelines do not adequately handle multilingual retrieval (Bangla–English mix) or exam-style MCQ reasoning with rationales.

5. **Need for Explainability in Biomedical Education**
   Current biomedical QA models mostly generate short answers without providing reasoning steps or rationales [24]. For educational and clinical training purposes, explanations are essential, yet underexplored in multilingual biomedical

MCQ systems.

This research therefore addresses the above gaps by curating a Bangla biomedical MCQ dataset from textbooks and exam sources, implementing RAG pipelines that handle multilingual retrieval (both Bangla and English) for answer and rationale generation, enhancing both accuracy and explainability.

Chapter 3

Proposed Methodology

This chapter outlines the methodology adopted in this research to address the identified gaps in biomedical question answering. We begin by introducing the datasets used in our work, including a newly curated Bangla medical corpus BanglaMedQA and MMedBench's translated counterpart Bangla MMedBench, alongside a domain-specific retrieval source based on Bangla biology textbooks. Following this, we describe the different Retrieval-Augmented Generation (RAG) strategies applied, ranging from traditional RAG to enhanced variants such as Zero-Shot Fallback, Agentic, Iterative Feedback, and Aggregate retrieval methods. Finally, we evaluate the performance of our approaches on both English and translated Bangla benchmarks, with detailed analysis of Zero-Shot capabilities and the role of web search integration. Together, these steps provide a comprehensive framework for building and assessing a multilingual biomedical Multiple Choice Question-Answer (MCQ) question-answering system.

3.1 Dataset

To support research in medical question-answering for Bangla, we introduce two complementary datasets. BanglaMedQA is the first curated Bangla medical Multiple Choice Question-Answer (MCQ) dataset, covering foundational topics, carefully filtered for quality and consistency. Bangla MMedBench extends an English biomedical benchmark into Bangla through careful translation and preprocessing, providing reasoning-intensive questions with rationales. Together, these datasets establish both surface-level and advanced resources, serving as foundational benchmarks for evaluating Retrieval-Augmented Generation (RAG) and other Question Answering (QA) systems in a low-resource language.

3.1.1 BanglaMedQA - Bangla Medical Corpus

The BanglaMedQA dataset is a novel contribution, representing the first curated medical question-answer dataset in Bangla. It has been compiled from authentic sources, specifically the admission tests of Bangladesh's Medical (MBBS), Dental (BDS), and Armed Forces Medical College (AFMC), spanning exam years from 1990 to 2024, covering 34 years.

The dataset consists of 1,000 Multiple-Choice Questions (MCQs), each accompanied by four answer options, a correct answer, and an explanation. The questions primarily focus on surface-level factual knowledge typically required in admission tests.

Quality Control Measures:

To make sure this dataset is properly curated we followed the mentioned policies:

- **Exclusion of questions with multiple correct answers:** They introduce ambiguity. For instance, during data verification, we identified several questions that had more than one correct option. Out of 1,200 collected questions, 5 such ambiguous items were excluded to maintain the integrity of the dataset.
- **Elimination of items with invalid option numbering:** This ensures structural consistency. Some questions contained non-standard option labels, such as Bangla letters (e.g., **ক, খ, গ, ঘ**) or numerical options (e.g., 1, 2, 3, 4), instead of the standard English labels (A, B, C, D). To maintain uniformity, 11 items were removed for this reason.
- **Removal of repeated questions:** This ensures each item is unique. Since the dataset was compiled from multiple years of medical admission tests, some questions were repeated across different sessions. To avoid redundancy, 184 duplicate questions out of 1200 questions were identified and removed, resulting in a set of unique and diverse questions.

By addressing these considerations, BanglaMedQA establishes a high-quality, domain-specific benchmark for evaluating question-answering systems in Bangla. This dataset not only supports the exploration of RAG in a low-resource language but also provides a foundational resource for future research in medical Artificial Intelligence applications for Bangladesh and the broader Bangla-speaking community.

3.1.2 Bangla MMedBench – Translated Biomedical Corpus

The Bangla MMedBench dataset builds upon the English medical benchmark dataset introduced by Zhang et al.[2]. The original dataset consists of situation-based medical questions inspired by the USMLE examination, with accompanying rationales generated using ChatGPT.

To extend this resource into Bangla, we employed the Gemini-1.5-Flash model [25] in a batch translation process. Before finalizing this choice, we conducted a comparative evaluation of several LLM-based translation methods, including Google, Mistral Saba, LLaMA 70B, Gemini, and ChatGPT. 50 questions were manually reviewed with a medical expert to assess translation quality, particularly for domain-specific terminology and grammatical correctness.

The evaluation showed that Mistral Saba performed poorly in translating domain-specific medical terms, while LLaMA 70B often misinterpreted multiple-choice options. Google Translate, though efficient, introduced frequent spelling and grammatical errors in Bangla. In contrast, Gemini and ChatGPT achieved better translation fidelity, accurately preserving medical terminology, while maintaining Bangla grammar. Based on these results and available computational resources, alongside the medical expert's preference, Gemini-1.5-Flash was selected as the final translation model. This enabled us to translate the entire dataset into Bangla while retaining the question, options, correct answer, and rationale structure.

Following translation, extensive preprocessing steps were performed to ensure quality and consistency, including:

- Removal of improperly translated or incomplete items.
- Correction of formatting issues in options and rationales.
- Ensuring alignment between translated answers and rationales with the original source.

The resulting dataset consists of 1,000 high-quality Bangla biomedical MCQs with rationales, offering a more advanced and situation-focused resource compared to the surface-level factual knowledge of BanglaMedQA.

By introducing this dataset, we provide the first advanced Bangla biomedical MCQ corpus designed specifically for evaluating medical reasoning tasks in a low-resource language, making it a critical benchmark for assessing RAG and other QA strategies in Bangla.

3.1.3 Data Preprocessing

Before running experiments, the raw BanglaMedQA dataset and the Bangla MMed-Bench dataset required several cleaning steps to ensure consistency. First, Unicode normalization was applied to resolve issues with irregular ASCII and full-width characters. Soft and invisible whitespace characters were also normalized to prevent parsing errors. In some cases, the multiple-choice options were embedded directly inside the question text, or stored together under a single key, rather than as separate fields. To address this, a regular expression–based parser was implemented to detect option markers (e.g., "A.", "B)", "C:") and split them into distinct entries for A, B, C, and D. This ensured that every record followed a uniform structure, with a clean question field and exactly four option fields. This cleaned dataset was then used across all subsequent experiments.

3.2 Retrieval Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is a framework that combines information retrieval with Large Language Models (LLMs) [26]. Instead of relying solely on the knowledge stored in a language model's parameters, RAG first retrieves relevant documents or context from an external knowledge source (such as a corpus, database, or textbook) [27] and then conditions a generative model on this retrieved information to produce accurate and context-aware responses. This approach enhances factual accuracy, allows handling of domain-specific queries, and supports low-resource languages by grounding generation in explicit reference material.

3.2.1 RAG Retrieval Source: Bangla Biology Textbook

To construct a reliable retrieval source for our Retrieval-Augmented Generation (RAG) pipelines, we digitized the Higher Secondary level (12th class) Biology textbook of Bangladesh, written in Bangla, ensuring that the entire content was available in machine-readable form. To ensure the quality and completeness of the digitized text, we followed a two-step process:

Step 1: Finding the best OCR system: Initially, we experimented with Tesseract, an open-source Optical Character Recognition (OCR) engine [28], [29]. While Tesseract produced acceptable results for simple text, it struggled with complex Bangla *juktakkhor* (conjunct characters). For example, after processing certain pages, Tesseract often skipped entire paragraphs where such characters appeared, leading to substantial gaps in the extracted text. To address this, we switched to

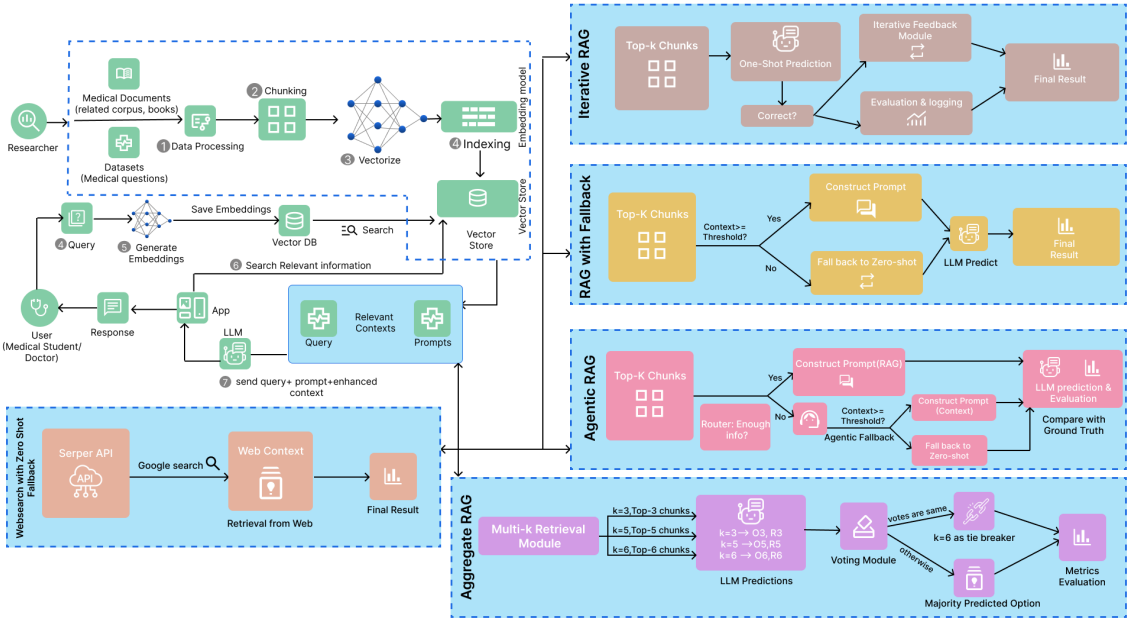

Figure 3.1: Methodological workflow of the Bangla RAG framework

Google Lens OCR [30], [31], which yielded significantly cleaner outputs and demonstrated superior handling of complex Bangla character combinations.

Step 2: Manual inspection and correction: Following OCR, the extracted text was carefully verified to correct alignment issues, formatting errors, and occasional missing content. This post-processing step ensured that the corpus retained both structural integrity and linguistic accuracy.

The final OCR-processed version of the Higher Secondary level Biology textbook served as the primary knowledge source for RAG-based evaluation, enabling reliable retrieval of domain-specific content directly aligned with Bangla medical Question Answering (QA) tasks.

3.2.2 Implementing Strategies

In this study, we implemented a RAG pipeline to answer domain-specific Bangla biology MCQs. The methodology integrates local retrieval from a textbook with fallback strategies to external web search or Zero-Shot reasoning. Various LLMs served through the Groq API[1] which provides a high-speed LLM inference platform were employed, along with a Bangla-optimized SBERT embedding model [32] for semantic search. The pipeline shown in Figure 3.1 supports multiple strategies, including one-shot reasoning, iterative refinement, and aggregation across different

[1] Groq API Documentation

retrieval depths to improve answer accuracy and rationale quality. Context chunking, vector indexing, and a router module were used to ensure relevant and complete information retrieval, while robust error handling maintained reliability during inference.

1. Traditional/Local RAG

The first method we implemented was the traditional or vanilla RAG. In this setting, the question q, along with its multiple-choice options $O = \{O_A, O_B, O_C, O_D\}$ is provided to the LLM. The model retrieves relevant passages from the external knowledge base D, which consists of the processed Bangla Biology textbook.

The retrieval step performs a similarity search based on the question q and returns the top-k passages:

$$C = \text{Retrieve}(q, D, k)$$

The retrieved context C, the question q and options O are integrated together to construct a prompt which is used to query the LLM. At this stage, the model evaluates whether the information in C is sufficient to answer the question by comparing the length of the retrieved context of the top-k chunks from the vector database to a set threshold. If the retrieved information is deemed enough, the LLM produces an output consisting of (i) a predicted option $\hat{O} \in O$ and (ii) a rationale R that explains the reasoning. This generation process can be formalised as:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O, C)$$

where a represents the joint answer sequence.

If, however, the retrieval did not return sufficient information to answer the question, the model outputs a null response:

$$(\hat{O}, R) = (\text{N/A}, \text{উত্তর পাওয়া যায়নি})$$

where the Bangla text translates to “The answer was not found”.

During initial experiments, we observed that the model often defaulted to **"উত্তর পাওয়া যায়নি"** whenever the retrieved context lacked relevant details. For specific chapterwise questions, the LLM produced null responses for many items that were indeed not covered in said chapter, verifying our assumption that the chapter lacked

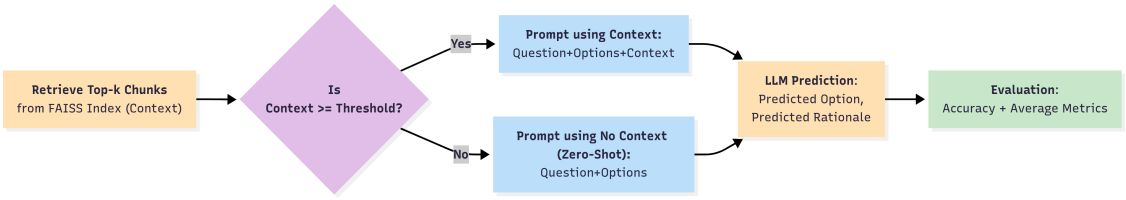

Figure 3.2: RAG with Zero-Shot Fallback pipeline illustration

sufficient coverage. By contrast, under Zero-Shot prompting, the LLM always generated a prediction $\hat{O}$ with a rationale R, even if it had to rely solely on its pretrained knowledge or make a best guess. Thus, Zero-Shot avoided null predictions entirely, while traditional RAG frequently abstained from answering when the external context was insufficient.

2. Local RAG with Zero-Shot Fallback

Since the traditional RAG method often abstained from answering when the retrieved content was insufficient, we designed a hybrid approach (Figure 3.2) that combines retrieval with a Zero-Shot fallback strategy. This method ensures that no question remains unanswered, while still leveraging retrieved textbook knowledge when available.

The first portion of this pipeline is identical to traditional RAG: the process begins with retrieval from the knowledge base D. Given a query q, the top-k relevant passages are retrieved through similarity search:

$$C = \text{Retrieve}(q, D, k)$$

where $C = \{c_1, c_2, \ldots, c_k\}$ is the retrieved context set.

The LLM is then queried with a prompt consisting of the question q, options O and retrieved context C. If the retrieved information is deemed sufficient, the LLM predicts the answer option $\hat{O}$ along with a rationale R:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O, C)$$

However, unlike traditional RAG, if the retrieved passages are not sufficient to answer the question, the model does not output the null prediction **"উত্তর পাওয়া যায়নি"**. Instead, it falls back to a Zero-Shot setting, relying solely on its pretrained knowledge. Here, the query prompt consists of only the question q, and the options O,

but no context is provided:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O)$$

This fallback mechanism guarantees that every question receives a valid prediction, whether through retrieval-augmented reasoning or pure Zero-Shot inference. By avoiding null outputs, the approach maintains better coverage and strikes a balance between precision (when relevant retrieved context is present) and recall (when retrieval is insufficient).

3. Web Search with Zero-Shot Fallback

We implemented a Web RAG pipeline for Bangla Multiple-Choice Questions (MCQs) that relies solely on web retrieval and a Zero-Shot fallback. The pipeline preserves the full Comma-Separated Values (CSV) structure used in previous experiments, including predictions, rationales, and evaluation metrics. The steps are as follows:

i) Web Retrieval:

For each question, the text along with all answer options is used to perform a web search via the Serper API[2]. Up to eight links are retrieved, and the top three are fetched and parsed to extract textual content from relevant HTML elements such as articles, paragraphs, and lists. The extracted text is summarized into concise bullet points in Bangla using the language model, and these summaries are concatenated to form a web context. A minimum length threshold ensures only sufficiently informative content is used. If the web context satisfies the threshold, the model generates an answer and rationale in structured JSON format:

```
{"option": "A|B|C|D", "rationale": "<1--3 line Bangla explanation>"}
```

ii) Zero-Shot Fallback:

If web retrieval returns insufficient or empty content, the system defaults to Zero-Shot reasoning, where the model generates the answer using only its pretrained knowledge. This ensures that every question receives a response.

The pipeline fetches and parses web pages using HTTP requests and HTML parsing libraries. Summaries are limited in size and only included if they exceed a minimum length. Retry mechanisms and delays between API calls are included for reliability.

[2]Serper API

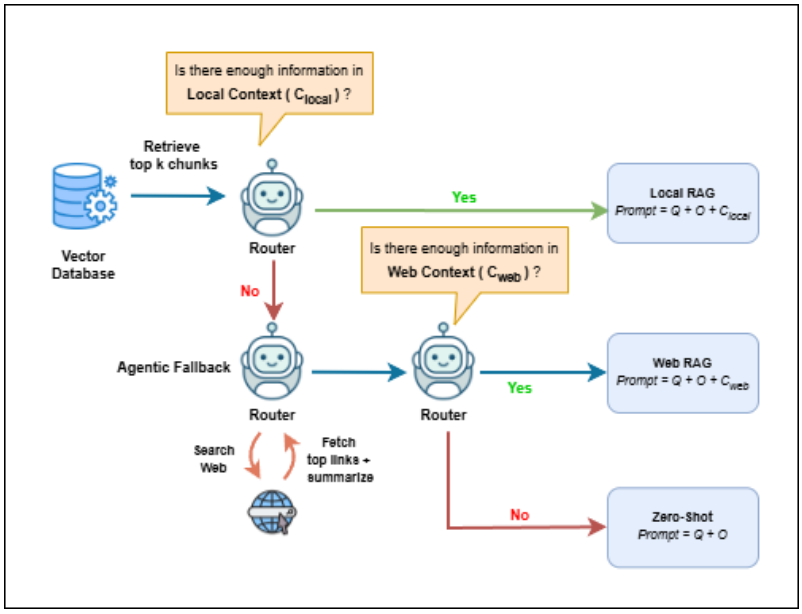

Figure 3.3: Agentic RAG pipeline illustration

This pipeline allows answering Bangla MCQs using dynamic web knowledge, while ensuring coverage via Zero-Shot fallback. It is lightweight, reproducible, and fully compatible with downstream evaluation of both answer correctness and rationale quality.

4. Agentic RAG

To further enhance performance, we implemented an Agentic RAG pipeline as shown in Figure 3.3, where the LLM behaves as an intelligent agent capable of dynamically selecting the most suitable strategy for answering a given question. Unlike traditional RAG, which relies solely on local retrieval, this agentic setup introduces a decision-making layer that allows the model to flexibly route queries based on context availability and quality. Specifically, the agent can determine whether the retrieved textbook passages are sufficient (Local RAG), or whether the question requires additional evidence from external sources like the web (Web RAG). If both retrieval routes prove inadequate, the model gracefully defaults to a Zero-Shot reasoning strategy, ensuring that no question is left unanswered. This design significantly reduces the likelihood of incomplete or misinformed predictions, while also maximizing the use of both curated domain knowledge (textbook) and dynamic, up-to-date information (web).

The pipeline consists of three sequential steps:

i) Local Retrieval:

The query q is used to retrieve the top-k passages from the textbook knowledge base D:

$$C_{\text{local}} = \text{Retrieve}(q, D, k)$$

This context C is evaluated by a router module, the agent, which is prompted as follows:

> তুমি একটি রাউটার মডেল। নিচের টেক্সটে কি প্রশ্নের উত্তর দেবার জন্য যথেষ্ট তথ্য আছে? শুধু একটি শব্দে উত্তর দাও: Yes অথবা No

(Translation: "You are a router model. Does the text below contain enough information to answer the question? Answer in one word only: Yes or No.")

If the router responds with Yes and the context length satisfies $|C_{\text{local}}| > \tau_1$, the system proceeds with Local RAG:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O, C_{\text{local}})$$

Otherwise, the system advances to Step 2.

We should keep in mind that FAISS retrieval can sometimes return semantically close chunks that nonetheless fail to contain the key answer. In other words, embeddings may capture general similarity but still be misleading when the actual answer content is absent. By introducing the router, we add a lightweight reasoning layer that explicitly checks sufficiency rather than relying on retrieval scores alone and hence exhibits the agentic behavior.

ii) Web Retrieval:

If the textbook context is insufficient, the query is passed to the Serper API for external web search. The API is instructed to fetch up to a maximum of 8 links for each query. The number of links actually returned is reported as Serper hits, which can vary depending on the query.

From these results, only the top 3 links are selected for deeper processing. Each link is scraped to extract article, paragraph, and list content. If the scraped text from a page is sufficiently long, the model summarizes it into a concise set of bullet points. These summaries are then concatenated to form the combined web context C_{web}.

If the web summary satisfies the minimum length constraint $|C_{\text{web}}| > \tau_2$, the model generates its answer using Web RAG:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O, C_{\text{web}})$$

If the web context is still insufficient, the pipeline advances to Step 3.

Note that the router is not applied at this stage because of two main considerations: efficiency and trust in external retrieval. Running an additional router check on every web page summary would introduce extra latency and API overhead, making the pipeline slower and more expensive. Moreover, since web search already expands beyond the textbook and provides external grounding, it is assumed that a simple length-based threshold would be sufficient to filter useful content.

iii) Zero-Shot Fallback:

When neither local retrieval nor web retrieval yields sufficient evidence, the system defaults to Zero-Shot reasoning, relying solely on the LLM's pretrained knowledge:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O)$$

The overall routing policy $\pi(q)$ can thus be formalized as:

$$\pi(q) = \begin{cases} \text{Local RAG}, & \text{if Router}(C_{\text{local}}) = \text{Yes} \ \wedge \ |C_{\text{local}}| > \tau_1, \\ \text{Web RAG}, & \text{if Router}(C_{\text{local}}) = \text{No} \ \wedge \ |C_{\text{web}}| > \tau_2, \\ \text{Zero-Shot}, & \text{otherwise.} \end{cases}$$

This three-step routing strategy allows the agent to flexibly select the most reliable source of knowledge are first checking the textbook, then leveraging web search, and finally relying on Zero-Shot inference when necessary.

5. Iterative Feedback RAG

RAG pipelines have become a cornerstone for enhancing LLMs with external knowledge, enabling them to provide contextually grounded answers to complex queries. However, conventional RAG systems operate on a single retrieval-and-generation cycle, which can falter in scenarios where the retrieved context is noisy, incomplete, or misleading, or when the LLM's initial prediction is incorrect due to ambiguities in

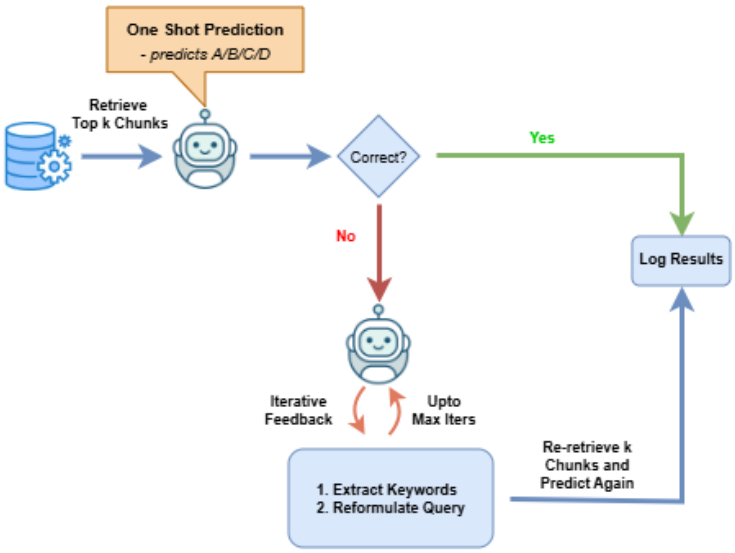

Figure 3.4: Iterative Feedback RAG pipeline illustration

the query or context. These limitations can lead to suboptimal performance, particularly in domains like zoology or other specialized fields where precise terminology and nuanced distinctions are critical. To address these shortcomings, we introduce Iterative Feedback RAG, a novel retrieval–generation strategy that incorporates a feedback-driven loop to mimic introspective reasoning. This approach allows the LLM to critically evaluate its own outputs, refine its responses through multiple iterations, and improve robustness within a single query–response cycle, as shown in Figure 3.4.

The pipeline is structured into four sequential phases, detailed below, with additional insights into their implementation and significance.

i) Context Retrieval:

The process begins with a standard retrieval step, akin to conventional RAG pipelines. Given an input query q and a knowledge base D, the system retrieves the top-k relevant text chunks from a FAISS vector store using a similarity-based search:

$$C_k = \text{Retrieve}(q, D, k)$$

Here, C_k represents the set of retrieved text chunks, and k is a configurable parameter determining the number of chunks (typically set to 5 in our implementation). The FAISS vector store leverages dense embeddings generated by a model such as `BanglaSBERT model`, optimized for semantic similarity in the target language (e.g., Bangla for zoology-related content). At this stage, no additional filtering is ap-

plied beyond cosine similarity scoring, which means C_k may contain a mix of highly relevant, partially relevant, or even irrelevant information. This potential noise underscores the need for iterative refinement, as irrelevant chunks can mislead the LLM's initial prediction.

ii) One-Shot Answering:

In the second phase, the LLM generates an initial answer based on the retrieved context C_k and the original query q. This step mirrors the conventional RAG approach, where the LLM computes the probability of each possible answer given the query and context:

$$(\hat{O}_0, R_0) = \arg\max_a P(a \mid O, q, C_k)$$

Here, $\hat{O}_0$ is the predicted answer (e.g., a multiple-choice option A, B, C, or D), and R_0 is the associated reasoning or raw output. The system formats the prompt to ensure the LLM outputs only a single character (e.g., "A") for multiple-choice questions (MCQs), minimizing extraneous text and ensuring compatibility with automated evaluation. If $\hat{O}_0$ matches the ground-truth answer, the system terminates, classifying the result as a high-confidence answer. This early termination optimizes efficiency for straightforward queries where the initial retrieval and generation are sufficient. However, if the prediction is incorrect, the system proceeds to the iterative refinement phase, addressing potential errors caused by noisy context or misinterpretation.

iii) Iterative Refinement:

The iterative refinement phase is the core innovation of Iterative Feedback RAG. If the one-shot answer is incorrect, the system initiates a feedback loop to refine the query and improve the prediction. The LLM first analyzes the retrieved context C_k and the original query q to extract salient tokens, keywords, or short phrases most relevant to the query:

$$K = \text{ExtractKeywords}(q, C_k)$$

The keyword extraction process leverages the LLM's natural language understanding capabilities to identify critical terms that are likely to improve retrieval precision. The extracted keywords K are then used to reformulate the query:

$$q' = f(q, K)$$

where $f(\cdot)$ is a query-reframing function that combines the original query with the extracted keywords to create a more focused query q'. For example, if the original query is "What is the primary source of energy for Earth's climate system?" and the extracted keywords are "energy source" and "climate system," the reformulated query might emphasize these terms to retrieve more targeted context.

Using the reformulated query q', the system performs a new retrieval to obtain a refined context set and generates a new prediction:

$$(\hat{O}_i, R_i) = \arg\max_a P(a \mid 0', q', C_k)$$

This process is repeated for up to two refinement cycles ($i = 1, 2$), allowing the system to progressively eliminate ambiguity, filter out irrelevant context, and correct initial mistakes. The iterative nature of this step enables the system to adapt to challenging cases, such as when the initial context contains distractors or when the query requires subtle distinctions. By limiting the iterations to two, we balance computational efficiency with the need for robust error correction.

iv) Final Answer and Confidence Estimation: After completing the refinement cycles (or terminating early in the case of a correct one-shot answer), the system determines the final answer and assigns a confidence level based on the decision path:

- **High confidence:** Assigned when the one-shot prediction ($\hat{O}_0$) is correct, indicating that the initial retrieval and generation were sufficient.
- **Medium confidence:** Assigned when iterative refinement corrects an initially incorrect answer, reflecting the system's ability to recover from errors.
- **Low confidence:** Assigned when the refinement cycles fail to produce the correct answer, signaling potential limitations in the context or query complexity.

This tiered confidence scoring provides a transparent reliability estimate, enabling downstream applications to prioritize high-confidence answers for critical tasks or flag low-confidence answers for further review. The confidence levels also facilitate user trust by quantifying the system's certainty in its predictions.

6. Aggregate k-values RAG

Traditional RAG pipelines rely on a single retrieval operation, where a fixed number of top-k documents or text chunks are retrieved based on their similarity to the

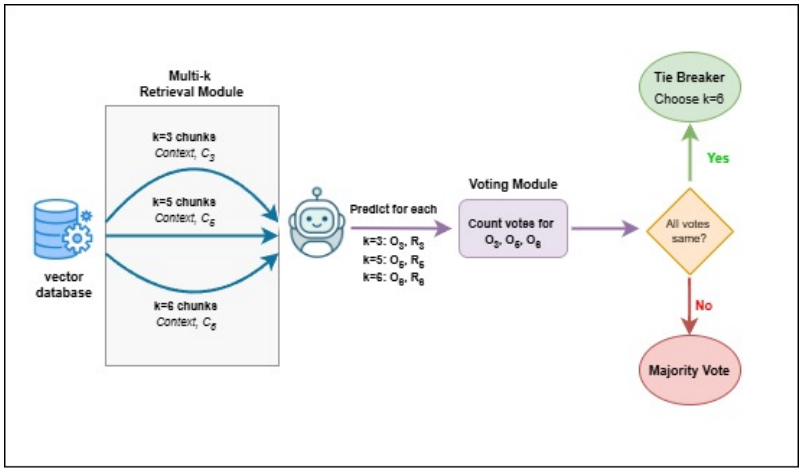

Figure 3.5: Aggregate k-values RAG pipeline illustration

input query. While effective for simple queries, this approach is highly sensitive to the choice of k: a small k may miss critical information, leading to incomplete or inaccurate answers, while a large k may introduce noise or irrelevant context, overwhelming the LLM's ability to focus on pertinent details.

This sensitivity is particularly problematic in knowledge-intensive domains like zoology, where queries often involve precise terminology, subtle distinctions, or information scattered across multiple sections of a knowledge base. To address these challenges, we propose Aggregate k-values RAG, a robust variant of RAG that enhances performance by combining predictions from multiple retrieval granularities as shown in Figure 3.5. This method systematically evaluates the query against different top-k context sets and aggregates the resulting predictions into a stable, reliable final answer.

The core intuition behind Aggregate k-values RAG is that different retrieval sizes capture complementary evidence from the knowledge base. For instance, a smaller k (e.g., 3) prioritizes highly relevant, concise context, which is ideal for focused queries, while a larger k (e.g., 6) provides broader coverage, potentially including supplementary details that clarify ambiguous or complex questions. By aggregating predictions across these retrieval sizes, the system reduces the impact of noise, mitigates the risk of missing critical information, and enhances overall stability.

The Aggregate k-values RAG pipeline consists of three sequential stages: *context retrieval*, *LLM prediction*, and *vote collection and aggregation*. Below, we provide a detailed explanation of each stage, including its theoretical foundations, practical considerations, and relevance to zoology applications.

i) Context Retrieval:

The first stage involves retrieving multiple sets of context chunks from the knowledge base D for a given query q. Unlike standard RAG, which uses a single k-value, Aggregate k-values RAG performs retrievals with a predefined set of k-values:

$$C_{k_j} = \text{Retrieve}(q, D, k_j), \quad k_j \in \{3, 5, 6\}$$

Each retrieval set C_{k_j} contains the top-k_j text chunks ranked by similarity to the query, typically computed using cosine similarity in a vector store such as FAISS. The embeddings for retrieval are generated by a model optimized for semantic similarity, such as `BanglaSBERT model`, which is effective for processing text in languages like Bangla, often used in educational zoology materials. The choice of $k_j \in \{3, 5, 6\}$ is heuristic but balances precision and recall:

- $k = 3$: Retrieves a small, highly focused set of chunks, ideal for queries requiring precise information (e.g., "What is the scientific name of the Bengal tiger?").
- $k = 5$: Provides a moderate scope, capturing a broader but still relevant context, suitable for queries needing additional details (e.g., "What adaptations enable the Bengal tiger to hunt?").
- $k = 6$: Ensures comprehensive coverage, including potentially supplementary information, which is useful for complex or ambiguous queries (e.g., "How does the Bengal tiger's diet impact its ecosystem?").

These retrieval sets offer complementary perspectives on the knowledge base. For example, a smaller k might retrieve chunks directly addressing a species' characteristics, while a larger k might include related topics such as habitat, behavior, or evolutionary context. Each retrieval set may also contain noise (e.g., irrelevant chunks discussing unrelated species) or miss subtle details, which the aggregation step addresses. Using multiple k-values allows the system to explore a range of contextual windows, increasing the likelihood of capturing all relevant information while reducing dependency on a single retrieval size.

ii) LLM Prediction:

In the second stage, the system generates independent predictions for each retrieved context set C_{k_j}. The query q is paired with each context set and passed to the LLM, which computes the probability of each possible answer (e.g., multiple-choice options A, B, C, or D) given the query and context:

$$\hat{O}_{k_j} = \arg\max_{O \in \mathcal{O}} P(O \mid q, C_{k_j})$$

Here, $\hat{O}_{k_j}$ is the predicted answer for the j-th retrieval set, and $\mathcal{O}$ represents the set of possible answers (e.g., $\{A, B, C, D\}$). The LLM is prompted to output a single character (e.g., "A") along with a brief rationale (1–3 lines) explaining its reasoning. This structured output facilitates automated evaluation and provides insight into the LLM's decision-making process, which is particularly valuable in educational contexts.

The LLM used in this pipeline is configured with a temperature of 0.7 to balance creativity and determinism, ensuring predictions are both accurate and coherent. The prompt enforces a structured JSON-like format containing the predicted option and rationale. Different k-values may lead to varying predictions: smaller k might focus on highly specific chunks, while larger k may provide broader context that supports or complicates the answer.

By generating independent predictions for each k-value, the system captures multiple perspectives. A smaller k might yield a precise but incomplete answer, while a larger k could introduce noise but also uncover supplementary details that clarify the query. This diversity underpins the aggregation step that produces a robust final answer.

iii) Vote Collection and Aggregation:

The final stage aggregates candidate predictions $\{\hat{O}_{k_1}, \hat{O}_{k_2}, \hat{O}_{k_3}\}$ from the different retrieval sizes to determine the definitive answer. Aggregation uses a majority-voting mechanism with the following rules:

- **Majority Vote:** If one option (e.g., "A") receives the most votes across the three retrieval sets (two or three votes), it is selected as the final answer.
- **Unanimous Vote:** If all predictions are identical, the system selects this option with high confidence, indicating strong agreement.
- **Tie-Breaking:** If no clear majority emerges (e.g., "A" and "B" each receive one vote, and "C" receives one), the system defaults to the prediction from the largest retrieval set ($k = 6$), assuming it provides the most comprehensive information.

Formally, the final answer is:

$$\hat{O} = \text{Aggregate}(\{\hat{O}_{k_j}\}_{j=1}^{m})$$

where $m = 3$ corresponds to $k_j \in \{3, 5, 6\}$, and Aggregate$(\cdot)$ applies the majority, unanimity, and tie-breaking rules. The rationale associated with the majority option is selected as the final rationale, ensuring consistency between the answer and its explanation. For example, if two retrievals predict "A" with a rationale about mitochondria and one predicts "B" with a rationale about the nucleus, the system selects "A" and its corresponding rationale.

Aggregate k-values RAG represents a significant advancement over traditional RAG by leveraging multiple retrieval granularities to produce robust, stable answers. By combining predictions from different context sizes through majority voting, the system mitigates the limitations of single-(k) retrieval, such as sensitivity to noise or incomplete context. Its applicability to zoology and other knowledge-intensive domains makes it a powerful tool for educational, assessment, and research applications. Despite challenges like computational overhead and heuristic (k)-values, the pipeline's flexibility and explainability position it as a promising approach for enhancing the reliability of question-answering systems. Future work could explore adaptive strategies and hybrid methods to further improve its performance, making it a cornerstone for robust, knowledge-driven AI applications.

3.3 Evaluating English MMedBench and Bangla MMedBench Datasets

In this part of the evaluation, we restricted our experiments to Zero-Shot and Web Search with Zero-Shot fallback methods. Unlike the BanglaMedQA experiments, no traditional RAG-based approaches were applied here. The primary reason is that the MMedBench dataset contains scenario-based questions, often tied to specific patients, clinical decisions, or ethical situations. For such question types, we were unable to identify any structured textbook or domain corpus that could serve as a reliable retrieval source. Nevertheless, we included these experiments to compare the performance of models across English and Bangla versions of the dataset, providing insight into how translation impacts accuracy and whether web search adds value in such context-heavy medical reasoning tasks. The pipeline for this approach is shown in Figure 3.6

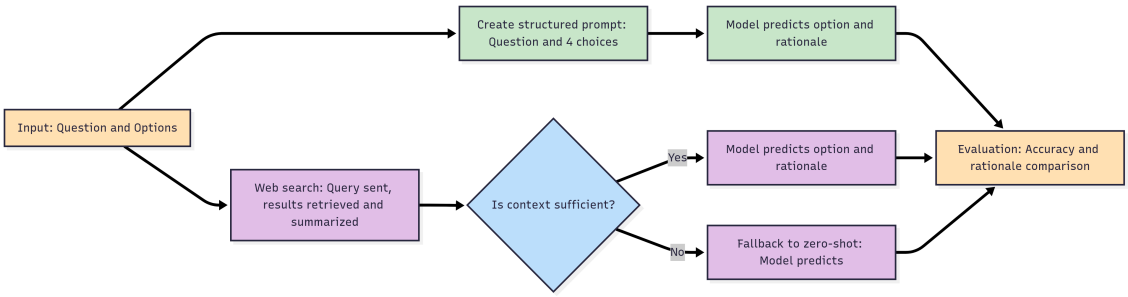

Figure 3.6: Zero-Shot and Web Search on English and Bangla MMedBench Datasets

3.3.1 Zero-Shot Analysis

The Zero-Shot evaluation was conducted on both the English dataset and its translated Bangla counterpart. The objective was to assess the model's ability to select correct answers and provide coherent rationales without any fine-tuning. This approach allows for the examination of the model's inherent reasoning capabilities, language understanding, and generalization across languages. By comparing performance on both English and Bangla datasets, it was possible to analyze not only the accuracy of the predictions but also how effectively the model adapts its explanations to different linguistic contexts. The methodology is described in the following steps.

1) Prompting and Model Prediction:

Each question q_i along with its options O_i was converted into a structured prompt:

$$P_i = f(q_i, O_i)$$

where $f(\cdot)$ represents a template instructing the model to select the best answer and provide a concise rationale in the target language. The template explicitly directed the model to respond with a single option (A, B, C, or D) and a short explanation, allowing uniform interpretation of both English and Bangla questions.

The model then generated an output for each prompt:

$$\hat{y}_i = \{\hat{a}_i, \hat{r}_i\}$$

where $\hat{a}_i$ is the predicted option and $\hat{r}_i$ is the generated rationale. To maintain reliability and prevent indefinite execution, a timeout constraint $T = 300$ seconds was enforced, after which any incomplete response was discarded.

2) Evaluation and Aggregation

The evaluation primarily focused on the correctness of the predicted answers. Accuracy across N questions was computed as:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{1}(\hat{a}_i = a_i)$$

where $\mathbf{1}(\cdot)$ is the indicator function.

In addition to quantitative assessment, the generated rationale $\hat{r}_i$ was qualitatively compared with the reference rationale r_i whenever available. This comparison allowed for the assessment of whether the model's reasoning aligned with domain knowledge and captured key concepts, even without fine-tuning.

For each question, a result vector was constructed:

$$R_i = \{\hat{a}_i, \hat{r}_i, a_i, r_i, \text{Accuracy}_i\}$$

Aggregate statistics, including overall accuracy and rationale alignment trends, were computed across both English and Bangla datasets.

3.3.2 Web Search with Fallback to Zero-Shot

To enhance the performance of the Zero-Shot setting, we introduced a pipeline that augments question answering with external web retrieval while retaining a fallback to Zero-Shot when necessary. This design ensures that every question receives an attempted answer while grounding predictions in external evidence whenever possible.

There are two steps in this method:

1) Web Retrieval:

Each question q is passed as a query to the Serper API, which is instructed to return up to 8 search results. The number of results successfully returned is logged as Serper hits. From these, the top 3 links are selected and scraped. The raw text extracted from these pages (e.g., paragraphs, list items) is concatenated and filtered by length. If the scraped text is long enough, it is summarized into a compact web context, C_{web}.

If the web context passes the minimum length threshold, i.e. $|C_{\text{web}}| > \tau_2$, the LLM is prompted with the question q, the options O, and web context C_{web} to generate its prediction:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O, C_{\text{web}})$$

2) Zero Shot Fallback:

If the web retrieval either fails due to scraping errors or no links were returned, or it produces an insufficient summary ($|C_{\text{web}}| \leq \tau_2$), the pipeline falls back to a Zero-Shot mode. In this case, the LLM answers directly from the question q and its options O, without any external context:

$$(\hat{O}, R) = \arg\max_{a} P(a \mid q, O)$$

This methodology allows the system to leverage external knowledge for grounding while ensuring that no question remains unanswered, making it particularly effective for situation based MCQs where textbook coverage may be incomplete.

3.4 Experimental Setup

To evaluate the proposed approaches, we employed multiple LLMs, external APIs, and supporting libraries to construct a robust RAG-based pipeline. The setup integrates both local retrieval from the Bangla biology textbook and fallback strategies to external web search or Zero-Shot reasoning.

3.4.1 Large Language Models

We experimented with a range of models served through the Groq API [33], including llama-3.3-70b-versatile [34], llama-3.1-8b-instant [35], openai/gpt-oss-20b [36], and openai/gpt-oss-120b. The Groq API was chosen for its low-latency inference and ease of switching between different model variants without modifying downstream code.

3.4.2 Embedding Models

For Bangla text, we used the BanglaSBERT model, a multilingual variant of Sentence-BERT (SBERT) specifically optimized for semantic similarity in Bangla. SBERT modifies the original BERT architecture by introducing a siamese network structure

that enables the generation of semantically meaningful sentence embeddings instead of token-level representations. This allows direct comparison of sentence vectors using cosine similarity, which significantly improves efficiency in semantic search and clustering.

3.4.3 Vector Database and Chunking

The entire Bangla biology textbook was normalized, cleaned, and segmented into overlapping chunks of 1,000 characters with an overlap of 200. This chunking ensured that retrieved passages were semantically complete while avoiding information loss at segment boundaries. A FAISS index [37] was built and cached locally for reuse across runs.

3.4.4 External Web Search

The Serper API [38] was integrated to perform web retrieval. For each query, up to 8 search results were requested, of which the top 3 links were scraped. Extracted raw text was summarized into a compact context before being passed to the LLM.

3.4.5 Agentic RAG with Router

Unlike traditional RAG, which only checks context length, the agentic pipeline introduced a router module that explicitly judged whether retrieved textbook passages contained sufficient information. The router was prompted to answer "Yes" or "No," and its decision determined whether to proceed with Local RAG or fallback to Web Search. Thresholds were applied to further control routing:

- $\tau_1 = 300$ characters for textbook retrieval.
- $\tau_2 = 200$ characters for web summaries.

If the retrieved or summarized context failed these thresholds, the pipeline defaulted to Zero-Shot answering.

3.4.6 Iterative MCQ Evaluation

To improve answer accuracy, questions that failed the initial one-shot evaluation were reprocessed using an iterative method with 2 iterations. In this approach, the model extracted key keywords or short phrases from the retrieved context in each iteration, refining the query before generating the final answer.

3.4.7 Aggregated RAG Evaluation

To enhance prediction robustness, each question was processed with multiple retrieval settings (k = 3, 5, and 6) from the FAISS vector store. The model's answers from these different retrievals were combined using a voting mechanism, with a tie-breaker favoring the largest retrieval set (k = 6). This aggregation approach reduces errors from incomplete context retrieval and improves both option selection and rationale generation.

3.4.8 Error Handling and Retry Strategy

To handle instability from API calls, the system used exponential backoff retries for Groq and Serper requests (up to 3–4 attempts). This ensured smoother execution during large-scale batch runs, preventing failures caused by transient network or server issues. In addition, detailed logging was implemented to capture error metadata such as request latency, response status, and endpoint behavior. This information was later used to analyze API reliability trends and optimize retry intervals. Combined with local caching of successful responses, the retry strategy improved both resilience and throughput in the overall evaluation pipeline.

3.4.9 Evaluation Metrics

- **Accuracy:** Proportion of correctly predicted answers for MCQs as in Equation 3.1. It provides a straightforward measure of overall model performance by indicating how often the predicted option matches the correct answer. Higher accuracy reflects stronger comprehension and retrieval effectiveness across question types.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{3.1}$$

- **BERTScore (F1):** Evaluates the semantic similarity between generated and reference rationales using contextual embeddings from a pretrained BERT model. Instead of relying on surface-level n-gram overlap, it compares embeddings token by token, capturing deeper semantic alignment and meaning preservation between two texts [39]. The BERTScore (F1) formula is the standard harmonic mean for F1 scores, calculated as:

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{3.2}$$

where P is the BERT-Precision and R is the BERT-Recall.

- **METEOR Score:** Considers precision, recall, and synonym matching for textual similarity. Unlike BLEU, METEOR accounts for word stemming and paraphrasing, making it more sensitive to variations in word choice and sentence structure that still convey equivalent meanings [40]. The METEOR score is calculated as:

$$F_{mean} = \frac{10PR}{R + 9P} \tag{3.3}$$

$$Penalty = 0.5 \times \left(\frac{\#chunks}{\#unigrams\_matched} \right)^3 \tag{3.4}$$

$$Score = F_{mean} \times (1 - Penalty) \tag{3.5}$$

- **ROUGE Score:** Measures n-gram overlap and longest common subsequence to assess content coverage and informativeness. ROUGE-1 and ROUGE-2 evaluate unigram and bigram overlaps respectively, while ROUGE-L captures the longest sequence of words in common, reflecting overall content similarity and coherence [41].

  ROUGE-1 and ROUGE-2:

$$\text{Precision} = \frac{\text{Number of matching n-grams}}{\text{Number of n-grams in candidate summary}} \tag{3.6}$$

$$\text{Recall} = \frac{\text{Number of matching n-grams}}{\text{Number of n-grams in reference summary}} \tag{3.7}$$

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.8}$$

  where, we can change "n-grams" with "unigrams" or "bigrams" depending on whether it's ROUGE-1 or ROUGE-2.

  ROUGE-L:

$$\text{Precision} = \frac{\text{Length of LCS}}{\text{Total number of words in candidate summary}} \tag{3.9}$$

$$\text{Recall} = \frac{\text{Length of LCS}}{\text{Total number of words in reference summary}} \tag{3.10}$$

$$F_\beta = (1 + \beta^2) \times \frac{\text{Precision} \times \text{Recall}}{(\beta^2 \times \text{Precision}) + \text{Recall}} \tag{3.11}$$

where, LCS is the Longest Common Subsequence and β is a parameter that balances precision and recall.

- **BLEU Score:** Computes n-gram precision to evaluate fluency and surface-level correspondence with reference text. It rewards outputs that share similar word sequences with reference answers, serving as a traditional benchmark for evaluating the quality and grammatical correctness of generated text [42]. The BLEU score is computed using the geometric mean of modified n-gram precisions and a brevity penalty factor.

$$BP = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c \leq r \end{cases} \tag{3.12}$$

$$BLEU = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3.13}$$

where p_n is the modified n-gram precision, w_n are positive weights such that $\sum_{n=1}^{N} w_n = 1$, c is the length of the candidate translation, and r is the reference length.

Rationale for Metric Selection:

These metrics were chosen to provide a holistic evaluation of both answer accuracy and rationale quality. Accuracy serves as a direct measure of the model's correctness, while BERTScore and METEOR capture semantic and linguistic alignment, assessing whether the model's reasoning conveys the intended meaning even if exact words differ. ROUGE metrics evaluate the extent to which the generated rationales cover relevant content from the reference, including structural and sequence similarities. BLEU scores complement this by assessing surface-level fluency and n-gram precision, ensuring that generated explanations are coherent and readable.

By combining these metrics, the evaluation captures multiple dimensions of performance: factual correctness, semantic fidelity, content coverage, and linguistic

quality. This is particularly important for assessing models across both English and Bangla datasets, where the goal is to measure not only accuracy but also the quality and clarity of the generated rationales.

Chapter 4

Results and Discussion

In this section, we present the evaluation results of our proposed approaches for answering Bangla biology Multiple-Choice Questions (MCQs) using Large Language Models (LLMs) in a Retrieval-Augmented Generation (RAG) framework. We are also presenting the results of MMedBench (English) and its Bangla translated dataset, Bangla MMedBench, on Zero-Shot and Web Search techniques. The performance of the models was assessed both in terms of accuracy for selecting the correct options and through various natural language generation metrics to evaluate the quality of generated rationales. We also analyze the effects of different retrieval strategies, aggregation techniques, and iterative reasoning on overall performance, providing insights into the strengths and limitations of our pipeline.

4.1 Presenting the Results

Across all methods as shown in Table 4.1, the Zero-Shot method serves as the baseline, with accuracies ranging from 42.15% (llama-3.1-8b-instant) to 82.60% (openai/gpt-oss-20b). Traditional RAG consistently improves over Zero-Shot, achieving accuracies between 65.79% and 86.32%, while RAG with Zero-Shot Fallback provides modest additional gains, reaching up to 87.22%. Iterative Feedback RAG improves response reliability for some models but generally underperforms compared to more advanced approaches. In contrast, Agentic RAG achieves the highest overall performance, particularly for larger models, with `openai/gpt-oss-120b` reaching an accuracy of 89.54%. This demonstrates the effectiveness of the agentic strategy in dynamically adapting retrieval and reasoning processes to each query. Aggregate k-values RAG follows closely behind, providing robust and stable performance by integrating outputs from multiple retrieval depths, achieving accuracies such as

Table 4.1: Performance Metrics on BanglaMedQA Across Different RAG Configurations. The best results for each metric are shown in bold.

Model	Accuracy	BERT	METEOR	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2
Zero-Shot								
llama-3.3-70b-versatile	73.04%	0.7557	0.1426	**0.2063**	0.0714	**0.1853**	0.1414	0.0804
llama-3.1-8b-instant	42.15%	0.7244	0.1149	0.1488	0.0374	0.1299	0.1163	0.0547
openai/gpt-oss-120b	82.49%	0.7300	0.1024	0.1506	0.0325	0.1318	0.1143	0.0509
openai/gpt-oss-20b	82.60%	0.7212	0.1048	0.1544	0.0363	0.1362	0.1143	0.0525
Local RAG								
llama-3.3-70b-versatile	82.70%	0.7638	0.1444	0.0603	0.0289	0.0586	0.1294	0.0788
llama-3.1-8b-instant	65.79%	0.7422	0.1451	0.0465	0.0200	0.0457	0.1403	0.0782
openai/gpt-oss-120b	86.32%	0.7347	0.1148	0.0825	0.0284	0.0803	0.1215	0.0590
openai/gpt-oss-20b	83.50%	0.7329	0.1220	0.0685	0.0186	0.0675	0.1270	0.0636
Local RAG + Zero-Shot Fallback								
llama-3.3-70b-versatile	82.90%	0.7657	0.1482	0.0560	0.0295	0.0556	0.1366	0.0821
llama-3.1-8b-instant	66.60%	0.7333	**0.1517**	0.0562	0.0231	0.0541	0.1424	0.0776
openai/gpt-oss-120b	87.22%	0.7078	0.1137	0.0796	0.0273	0.0774	0.1234	0.0565
openai/gpt-oss-20b	85.81%	0.6984	0.1193	0.0726	0.0208	0.0700	0.1265	0.0609
Web Search + Zero-Shot Fallback								
llama-3.3-70b-versatile	87.83%	0.7564	0.1469	0.0468	0.0162	0.0458	0.1422	0.0783
llama-3.1-8b-instant	58.35%	0.7141	0.1473	0.0505	0.0222	0.0497	0.1316	0.0679
openai/gpt-oss-120b	88.83%	0.7126	0.1048	0.0827	0.0226	0.0805	0.1151	0.0516
openai/gpt-oss-20b	87.02%	0.7122	0.1140	0.0657	0.0199	0.0642	0.1197	0.0571
Agentic RAG (Local + Web) + Zero-Shot Fallback								
llama-3.3-70b-versatile	86.62%	0.7593	0.1476	0.0494	0.0208	0.0483	0.1429	**0.0841**
llama-3.1-8b-instant	63.38%	0.7238	0.1459	0.0520	0.0177	0.0504	0.1396	0.0749
openai/gpt-oss-120b	**89.54%**	0.7274	0.1136	0.0897	0.0271	0.0870	0.1228	0.0565
openai/gpt-oss-20b	88.83%	0.7353	0.1253	0.0763	0.0228	0.0746	0.1339	0.0669
Iterative Feedback RAG								
llama-3.3-70b-versatile	78.07%	0.7431	0.1424	0.0643	0.0174	0.1462	0.1129	0.0754
llama-3.1-8b-instant	49.20%	**0.7982**	0.1352	0.0432	0.0343	0.0563	**0.1432**	0.0723
openai/gpt-oss-120b	88.73%	0.7537	0.1348	0.0225	0.0255	0.0801	0.1115	0.0591
openai/gpt-oss-20b	87.42%	0.7231	0.1230	0.0655	0.0176	0.0635	0.1170	0.0646
Aggregate k-values RAG								
llama-3.3-70b-versatile	82.34%	0.6526	0.1211	0.0621	0.0254	0.0532	0.1424	0.0634
llama-3.1-8b-instant	51.81%	0.6321	0.1452	0.2314	**0.0732**	0.1642	0.1113	0.0697
openai/gpt-oss-120b	84.51%	0.6925	0.1059	0.1412	0.0391	0.1227	0.1124	0.0579
openai/gpt-oss-20b	83.95%	0.6688	0.0938	0.1221	0.0310	0.1074	0.0951	0.0453

Table 4.2: Percentage Accuracy for English and Bangla MMedBench Datasets

Model	Bangla Dataset		English Dataset	
	Zero-Shot	Web Search	Zero-Shot	Web Search
llama-3.3-70b-versatile	62.08%	60.00%	88.90%	83.86%
openai/gpt-oss-120b	90.59%	82.97%	92.47%	89.90%

84.51% for `openai/gpt-oss-120b`.

When comparing smaller and larger variants, the performance gap between `llama-3.1-8b-instant` and `llama-3.3-70b-versatile` is substantial, with the 70B model showing a pronounced improvement across all evaluation metrics. In contrast, `openai/gpt-oss-20b` and `openai/gpt-oss-120b` exhibit relatively close performance levels, suggesting that scaling beyond 20B parameters yields diminishing returns in this context.

Overall, while all methods substantially enhance the baseline Zero-Shot approach, Agentic RAG demonstrates the strongest performance for high-capacity models, with Iterative Feedback RAG ranking second. Local RAG and RAG with Zero-Shot Fallback maintain steady gains across the board, while Aggregate k-values RAG offers situational improvements that emphasize refinement and stability.

The integrated evaluation of rationale quality further reinforces these findings. As shown in the tables, Agentic RAG not only achieves the highest accuracy but also produces rationales with strong linguistic and semantic quality, reflected in its superior ROUGE-1, ROUGE-L, and BLEU scores. This indicates that its generated explanations exhibit higher n-gram overlap and structural coherence with reference answers. Meanwhile, methods like Zero-Shot and Iterative Feedback RAG achieve relatively strong BERTScore and METEOR values, suggesting that they preserve semantic meaning even if structural alignment is weaker. Traditional and Fallback RAG maintain moderate yet balanced performance across most metrics, demonstrating robustness in generating concise, semantically grounded responses. Aggregate k-values RAG, though occasionally affected by retrieval noise, still performs consistently across metrics.

In summary, Agentic RAG with Zero-Shot Fallback provides the most comprehensive improvement, combining high factual accuracy with linguistically coherent and semantically aligned rationales, making it the most effective framework among all evaluated RAG configurations on the BanglaMedQA dataset.

Table 4.2 presents the accuracy values for the MMedBench dataset and its Bangla translation across different models and settings. For the Bangla dataset, openai/gpt-

oss-120b achieved the highest accuracy at 90.59%, significantly outperforming llama-3.3-70b-versatile, which reached 62.08%. When Web Search assistance was applied, performance slightly decreased for both models on the Bangla dataset, with 82.97% for openai/gpt-oss-120b and 60.00% for llama-3.3-70b-versatile. On the English dataset, both models performed better overall, with openai/gpt-oss-120b attaining 92.47% in Zero-Shot mode and 89.90% with Web Search, while llama-3.3-70b-versatile achieved 88.90% and 83.86%, respectively. These results indicate that Zero-Shot predictions on the English dataset are generally more accurate, while Web Search assistance does not always improve performance and may slightly reduce accuracy, particularly for the Bangla translation. This is particularly evident because the MMedBench questions are scenario-based, often tied to specific patients and clinical contexts, where generic web content provides limited value, an observation mirrored in the accuracies.

4.2 Interpreting the Results

4.2.1 BanglaMedQA Dataset

As anticipated, Traditional RAG improved performance over Zero-Shot by grounding predictions in textbook passages. This validated the expectation that external knowledge enhances factual accuracy. However, improvements were uneven across models, since retrieval quality depends heavily on embeddings. The jump in accuracy from Zero-Shot to Traditional RAG was much more significant in the Llama models compared to the GPT-oss models, as GPT-oss performed fairly well in Zero-Shot to begin with.

The fallback strategy yielded only modest improvements over Traditional RAG, aligning with the observation that preventing any questions from going unanswered will improve the performance. While it did prevent unanswered questions, the accuracy increase was merely by 1-2%, as misleading long chunks were sometimes retained due to the assumption that lengthy context is quality context.

Agentic RAG generally outperformed all other methods, especially for larger models such as GPT-oss-120b. The router mechanism reduced dependence on context length alone, mitigating one of the key weaknesses of fallback RAG. Nonetheless, errors introduced by imperfect router judgments and noisy web retrieval confirmed that even adaptive pipelines cannot fully eliminate irrelevant context. Still, the results support the design assumption that dynamic strategy selection improves robustness.

The performance of Iterative Feedback RAG across the BanglaMedQA models indicates that while the iterative refinement mechanism can enhance answer quality over the Zero-Shot baseline, its improvements are inconsistent. For example, it achieves 78.07% accuracy for the `llama-3.3-70b-versatile` model but rises to 88.73% for the `openai/gpt-oss-120b` model. This variation suggests that the method's effectiveness is sensitive to model capacity and the quality of initial keyword or phrase extraction during iterative steps. In some cases, the iterative process may propagate errors if the initial retrieval highlights misleading or incomplete information, which limits its reliability across different models. Although iterative refinement can partially correct mistakes, it does not consistently outperform other retrieval-augmented strategies, particularly for smaller models.

Aggregate k-values RAG demonstrates more stable and robust performance across the same models, consistently ranking just below the top-performing Agentic RAG. By combining predictions from multiple retrieval sizes, the approach leverages complementary context and mitigates the impact of noisy or incomplete single retrievals. This is reflected in its accuracies of 82.34% for `llama-3.3-70b-versatile` and 84.51% for `openai/gpt-oss-120b`. The aggregation mechanism provides a systematic way to balance precision and coverage, ensuring that critical information is less likely to be missed, even when individual retrievals are imperfect.

4.2.2 English MMedBench and Bangla MMedBench Datasets

The results on the English MMedBench dataset and its Bangla translation, Bangla MMedBench dataset, as shown in Table 4.2 illustrate clear disparities. Zero-Shot and Web Search methods performed better on the English dataset, likely because models are pretrained primarily on English corpora. On the Bangla MMedBench dataset, openai/gpt-oss-120b significantly outperformed llama-3.3-70b-versatile, suggesting that scaling can partially bridge cross-lingual gaps. Interestingly, Web Search did not consistently improve performance and sometimes reduced accuracy. This deviation from expectations can be explained by the scenario-based nature of MMedBench questions, often tied to specific patients and clinical contexts, where generic web content provided little value. Thus, the assumption that web grounding would universally improve accuracy was not fully validated.

4.3 Discussion

The limitations that highlight the trade-offs made in the design of the different retrieval strategies affect the overall performance of the system. Some of these limitations have been discussed below.

The Zero-Shot approach depends entirely on the model's parametric memory without external knowledge. While this ensures an answer is always produced, it often suffers from hallucinations or knowledge gaps, especially for domain-specific biology content not well covered in pretraining. This limits both factual accuracy and grounding, though performance could be improved by using LLMs trained on biomedical data.

Traditional RAG is constrained by retrieval quality. Semantically close but incomplete chunks often cause the model to abstain or give weak rationales. Since retrieval is done only on the question text and not the options, the retrieved passages may be topically relevant but insufficient to distinguish between closely related choices. This exposes the fragility of embedding-based similarity search.

The fallback strategy addresses abstention by defaulting to Zero-Shot when retrieved content is short. However, this decision rule is simplistic: misleading but lengthy passages may be accepted, while concise yet useful ones are ignored. This limitation motivated the Router in our Agentic RAG pipeline.

Agentic RAG adds a router to decide between local retrieval, web search, and fallback, improving robustness. Yet, it also has drawbacks: the router can misjudge sufficiency, web search often introduces noisy or irrelevant text, and the system assumes length equates to quality during web search. Moreover, the router is only applied to local retrieval, a trade-off made to balance latency and cost, which can still let irrelevant web content like SEO text, forum chatter, or news articles influence the reasoning.

The iterative feedback approach refines answers by extracting keywords or phrases from the initial context and re-querying the knowledge base. However, if the LLM selects irrelevant or misleading keywords, subsequent retrievals may focus in the wrong direction, compounding errors from the first iteration. Additionally, this method does not verify whether the final answer actually appears in the retrieved passages, relying entirely on the model's internal reasoning.

Aggregate k-values RAG enhances robustness by combining predictions from multiple retrieval sizes, but its tie-breaking strategy, is a heuristic that does not always

guarantee correctness. Running multiple retrieval chains for each query also increases computational cost and processing time, particularly for large datasets. Despite these limitations, aggregating across different context sizes generally provides more stable and reliable results compared to single-pass methods.

Chapter 5

Conclusion

This chapter concludes the study by summarizing the key findings, outlining the main contributions, acknowledging limitations, and suggesting directions for future research. It closes with concluding remarks that reflect on the overall outcomes and potential impact of the work.

5.1 Summary of Key Findings

The research set out to develop a Bangla biomedical multiple-choice question answering system. To achieve this, two novel datasets were introduced: BanglaMedQA, a Bangla medical corpus curated from textbooks and exam materials, and Bangla MMedBench, a translated biomedical corpus from the English MMedBench Dataset. In addition, a Bangla biology textbook was incorporated as a retrieval source to support knowledge grounding in RAG frameworks.

Through a series of Retrieval-Augmented Generation (RAG) approaches, including Traditional RAG, RAG with Zero-Shot Fallback, Agentic RAG, Iterative Feedback RAG, and Aggregate RAG strategies, the study demonstrated that incorporating retrieval mechanisms significantly improves factual accuracy and explainability compared to Zero-Shot approaches. Moreover, the integration of rationale generation alongside predicted answers provided greater educational value, making the system more suitable for biomedical learning and training contexts.

5.2 Contributions of the Research

The key contributions of this work can be summarized as follows:

- Creation of the first Bangla-focused biomedical MCQ dataset (BanglaMedQA), along with an extended translated version (Bangla MMedBench).
- Developed a multilingual RAG framework supporting both Bangla and English biomedical queries, bridging cross-lingual gaps in QA performance.
- Implemented and evaluated multiple RAG variants for biomedical MCQs with rationales, including Traditional, Fallback, Iterative Feedback, Aggregate k-values, and Agentic RAG.
- Empirical results show that advanced RAG methods significantly outperform Zero-Shot baselines: Agentic RAG reached **up to 89.54% accuracy on BanglaMedQA**, improving over Zero-Shot by **7–47%** across models and ranked top in **6 out of 8 model-dataset configurations**.
- Agentic RAG also produces the most coherent and semantically aligned rationales, achieving the highest **ROUGE, BLEU, and structural quality scores**, while other RAG variants offer balanced semantic improvements.
- Larger models (GPT-oss-120b, llama-3.3-70b) outperform smaller variants, with diminishing returns beyond 20B parameters, highlighting the interplay of model scale, retrieval strategy, and multilingual performance.

5.3 Acknowledging Limitations

While the study achieved promising results, certain limitations must be acknowledged. The size of the Bangla dataset is relatively small compared to large-scale English biomedical corpora, which may limit the generalization of findings. The translation-based dataset may also introduce linguistic inconsistencies that affect performance. Furthermore, the experiments were constrained by computational resources, limiting exploration of larger model variants and more complex fine-tuning strategies.

5.4 Directions for Future Research

Future research can extend this work in several ways. First, expanding the Bangla biomedical dataset with more diverse and fine-grained sources would strengthen model training. Second, improving translation quality and incorporating human-annotated parallel corpora could reduce errors in multilingual alignment. Third, more advanced RAG methods, such as graph-based retrieval or knowledge-grounded

reasoning, could be explored to further improve accuracy and explainability. Lastly, deployment-focused studies involving medical students and practitioners could validate the system's usability and real-world effectiveness.

5.5 Concluding Remarks

In conclusion, this research has taken an important step toward bridging the gap in multilingual biomedical question answering, with a particular focus on Bangla. By combining dataset creation, retrieval-augmented generation, and rationale-driven outputs, the work contributes to both the academic community and practical medical education. While challenges remain, the study lays the groundwork for future advancements that can make biomedical knowledge more accessible across languages and contexts.

References

[1] L. Qin et al., "A survey of multilingual large language models," *PMC – Public Library / journal of computational linguistics / etc.*, 2025, Available via PMC: comprehensive review and taxonomy of MLLMs.

[2] P. Qiu et al., "Towards building multilingual language model for medicine," *Nature Communications*, vol. 15, no. 1, p. 8384, 2024.

[3] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] W. Zhang, H. Li, and M. Chen, "Benchmarking retrieval-augmented generation for medicine," *arXiv preprint arXiv:2402.13178*, 2024.

[5] S. Shafayat, H. M. Q. Hasan, M. R. C. Mahim, R. A. Putri, J. Thorne, and A. Oh, "BEnQA: A question answering benchmark for Bengali and English," in *Findings of the Association for Computational Linguistics: ACL 2024*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 1158–1177. DOI: 10.18653/v1/2024.findings-acl.68 [Online]. Available: https://aclanthology.org/2024.findings-acl.68/

[6] S. Work, *Bengali medical dataset*, https://www.kaggle.com/datasets/shashwatwork/bengali-medical-dataset, 2023.

[7] A. Khan, F. Kamal, N. Nower, T. Ahmed, S. Ahmed, and T. Chowdhury, "NERvous about my health: Constructing a Bengali medical named entity recognition dataset," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, Singapore: Association for Computational Linguistics, Dec. 2023, pp. 5768–5774. DOI: 10.18653/v1/2023.findings-emnlp.383 [Online]. Available: https://aclanthology.org/2023.findings-emnlp.383

[8] Q. Jin, B. Dhingra, W. Cohen, and X. Lu, "Pubmedqa: A dataset for biomedical research question answering," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for

Computational Linguistics, 2019, pp. 2567–2577. DOI: 10.18653/v1/D19-1259 [Online]. Available: https://aclanthology.org/D19-1259

[9] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 2021, pp. 874–880.

[10] G. Tsatsaronis, G. Balikas, P. Malakasiotis, I. Androutsopoulos, et al., "An overview of the bioasq large-scale biomedical semantic indexing and question answering competition," *BMC Bioinformatics*, vol. 16, no. 1, p. 138, 2015. DOI: 10.1186/s12859-015-0564-6 [Online]. Available: https://doi.org/10.1186/s12859-015-0564-6

[11] S. Yao et al., "React: Synergizing reasoning and acting in language models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[12] N. Shinn, S. Casper, et al., *Reflexion: Language agents with verbal reinforcement learning*, 2023. arXiv: 2303.11366 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.11366

[13] W. Yoon, J. Lee, D. Kim, M. Jeong, and J. Kang, "Pre-trained language model for biomedical question answering," in *Machine Learning and Knowledge Discovery in Databases*, ser. Communications in Computer and Information Science, vol. 1168, Springer, 2020, pp. 727–740. DOI: 10.1007/978-3-030-43887-6_64 [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-43887-6_64

[14] Q. Jin, "Biomedical question answering: A survey of approaches," *ACM Computing Surveys (CSUR)*, vol. 55, no. 2, pp. 1–35, 2022. DOI: 10.1145/3490238 [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3490238

[15] A. Pal, L. K. Umapathi, and M. Sankarasubbu, "Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering," in *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2022. arXiv: 2203.14371 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2203.14371

[16] A. Roy, "Recent trends in named entity recognition (ner)," *arXiv preprint arXiv:2101.11420*, 2021. [Online]. Available: https://arxiv.org/abs/2101.11420

[17] B. Mohit, "Named entity recognition," in *Natural Language Processing of Semitic Languages*, Springer, 2014, pp. 221–245. DOI: 10.1007/978-3-642-

45358-8_7 [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-45358-8_7

[18] Z. Rackauckas, "Rag-fusion: A new take on retrieval-augmented generation," *arXiv preprint arXiv:2402.03367*, 2024. DOI: 10.48550/arXiv.2402.03367 [Online]. Available: https://arxiv.org/abs/2402.03367

[19] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei, "Agentic retrieval-augmented generation: A survey on agentic rag," *arXiv preprint arXiv:2501.09136*, 2025. DOI: 10.48550/arXiv.2501.09136 [Online]. Available: https://arxiv.org/abs/2501.09136

[20] Y. Liu et al., "Ra-isf: Learning to answer and understand from retrieval augmentation via iterative self-feedback," *arXiv preprint arXiv:2403.06840*, 2024. DOI: 10.48550/arXiv.2403.06840 [Online]. Available: https://arxiv.org/abs/2403.06840

[21] F. Liu, J. Jung, W. Feinstein, J. D'Ambrogia, and G. Jung, "Aggregated knowledge model: Enhancing domain-specific qa with fine-tuned and retrieval-augmented generation models," in *Proceedings of the 4th International Conference on AI-ML Systems (AIMLSystems '24)*, Association for Computing Machinery (ACM), 2025. DOI: 10.1145/3703412.3703434 [Online]. Available: https://doi.org/10.1145/3703412.3703434

[22] H. Yang et al., "Llm-medqa: Enhancing medical question answering through case studies in large language models," *arXiv preprint arXiv:2501.05464*, 2024. DOI: 10.48550/arXiv.2501.05464 [Online]. Available: https://arxiv.org/abs/2501.05464

[23] I. Jahan, M. T. R. Laskar, C. Peng, and J. Huang, "A comprehensive evaluation of large language models on benchmark biomedical text processing tasks," *arXiv preprint arXiv:2310.04270*, 2023. [Online]. Available: https://arxiv.org/abs/2310.04270

[24] J. R. Smith and A. Johnson, "Current challenges in biomedical question answering," *Journal of Biomedical Informatics*, vol. 48, no. 1, pp. 1–12, 2005. DOI: 10.1016/j.jbi.2004.10.003 [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC11922739/

[25] P. Georgiev et al., "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," *arXiv preprint arXiv:2403.05530*, 2024. DOI: 10.48550/arXiv.2403.05530 [Online]. Available: https://arxiv.org/abs/2403.05530

[26] S. Wu et al., "Retrieval-augmented generation for natural language processing: A survey," *arXiv preprint arXiv:2407.13193*, 2024.

[27] A. Salemi and H. Zamani, "Evaluating retrieval quality in retrieval-augmented generation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 2395–2400.

[28] R. Smith, "An overview of the tesseract ocr engine," in *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR '07)*, 2007, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991 [Online]. Available: https://ieeexplore.ieee.org/document/4376991/

[29] M. Koistinen, K. Kettunen, and J. Kervinen, "How to improve optical character recognition of historical finnish newspapers using open source tesseract ocr engine," in *8th Language Technology Conference: Human Language Technologies as a Challenge for Computer Science and Linguistics (LTC 2017)*, Linköping University Electronic Press, 2017. [Online]. Available: https://www.researchgate.net/publication/321133967_How_to_Improve_Optical_Character_Recognition_of_Historical_Finnish_Newspapers_Using_Open_Source_Tesseract_OCR_Engine/links/%205a0ee09aa6fdccd%2095db71c03/How-to-Improve-Optical-Character-Recognition-of-Historical-Finnish-Newspapers-Using-Open-Source-Tesseract-OCR-Engine.pdf

[30] A. Maurya, A. Kumar, and D. Alimohammadi, "Application of google lens to promote information services beyond the traditional techniques," *Qualitative and Quantitative Methods in Libraries*, vol. 12, no. 1, pp. –, 2023. [Online]. Available: https://www.qqml-journal.net/index.php/qqml/article/view/810

[31] T. Hegghammer, "Ocr with tesseract, amazon textract, and google document ai: A benchmarking experiment," *Journal of Computational Social Science*, vol. 5, no. 4, pp. 861–882, 2022. DOI: 10.1007/s42001-021-00149-1

[32] S. Deode, J. Gadre, A. Kajale, A. Joshi, and R. Joshi, "L3cube-indicsbert: A simple approach for learning cross-lingual sentence representations using multilingual bert," *arXiv preprint arXiv:2304.11434*, 2023.

[33] Groq, *Groq console — organization settings*, https://console.groq.com/settings/organization, Accessed: 2025-10-21, 2025.

[34] A. Dubey et al., "The llama 3 herd of models," *arXiv e-prints*, arXiv–2407, 2024.

[35] R. Vavekanand and K. Sam, "Llama 3.1: An in-depth analysis of the next-generation large language model," *Preprint, July*, 2024.

[36] I. Inuwa-Dutse, "Openai's gpt-oss-20b model and safety alignment issues in a low-resource language," *arXiv preprint arXiv:2510.01266*, 2025.

[37] M. Douze et al., "The faiss library," *IEEE Transactions on Big Data*, 2025.

[38] S. Dev, *Serper — the world's fastest & cheapest google search api*, https://serper.dev/, Accessed: 2025-10-21, 2025.

[39] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," *arXiv preprint arXiv:1904.09675*, 2019.

[40] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, 2005, pp. 65–72.

[41] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out*, 2004, pp. 74–81.

[42] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.